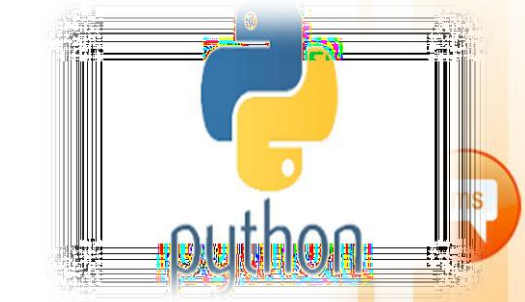
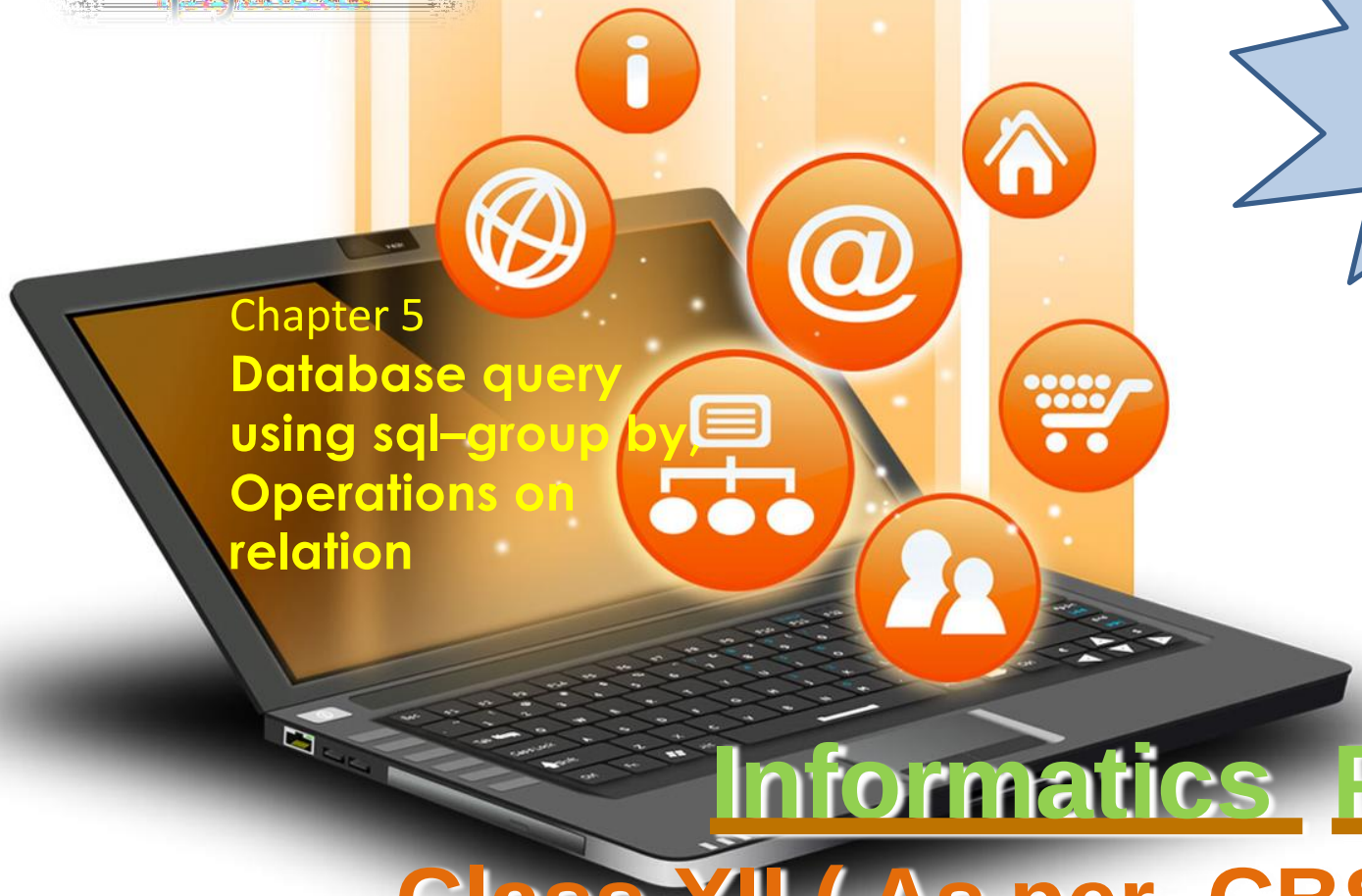


New syllabus 2021-22



Chapter 5

Database query using sql–group by, Operations on relation

Informatics Practices

Class XII (As per CBSE Board)

Visit : python.mykvs.in for regular updates

MySQL Order By

MySQL Order By clause is used to sort the table data in either Ascending order or Descending order. By default, data is not inserted into Tables in any order unless we have an index.

So, If we want to retrieve the data in any particular order, we have to sort it by using MySQL Order By statement.

Syntax:-SELECT Column_Names

FROM Table_Name

ORDER BY {Column1}[ASC | DESC] {Column2}[ASC | DESC]

MySQL Order By

MySQL Order by– e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+---------+---------+--------+---------+
| rollno  | name    | class  | marks   |
+---------+---------+--------+---------+
|       1 | freya   |     10 |      88 |
|       2 | mohak   |      1 |      99 |
|       3 | vishal  |     10 |      84 |
|       4 | vimal   |     10 |      82 |
|       5 | anil    |      2 |      82 |
+---------+---------+--------+---------+
```

Now we write the query – select * from student order by class;

```
mysql> select * from student order by class;
+---------+---------+--------+---------+
| rollno  | name    | class  | marks   |
+---------+---------+--------+---------+
|       2 | mohak   |      1 |      99 |
|       5 | anil    |      2 |      82 |
|       1 | freya   |     10 |      88 |
|       3 | vishal  |     10 |      84 |
|       4 | vimal   |     10 |      82 |
+---------+---------+--------+---------+
```

Query result will be in ascending order of class.If we not specify asc/desc in query then ascending clause is applied by default

MySQL Order By

MySQL Order by– e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+----------+----------+---------+----------+
| rollno   | name     | class   | marks    |
+----------+----------+---------+----------+
|        1 | freya    |      10 |       88 |
|        2 | mohak    |       1 |       99 |
|        3 | vishal   |      10 |       84 |
|        4 | vimal    |      10 |       82 |
|        5 | anil     |       2 |       82 |
+----------+----------+---------+----------+
```

Now we write the query – select * from student order by class desc;

```
mysql> select * from student order by class desc;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
|      2 | mohak  |     1 |    99 |
+--------+--------+-------+-------+
```

Query result will be in descending order of class

MySQL Order By

MySQL Order by – e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
+--------+--------+-------+-------+
```

Now we write query–select * from student order by class asc, marks asc;

```
mysql> select * from student order by class asc,marks asc;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      2 | mohak  |     1 |    99 |
|      5 | anil   |     2 |    82 |
|      4 | vimal  |    10 |    82 |
|      3 | vishal |    10 |    84 |
|      1 | freya  |    10 |    88 |
+--------+--------+-------+-------+
```

Query result will be ascending order of class and if same class exists then ordering will done on marks column(ascending order)

MySQL Order By

MySQL Order by– e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
+--------+--------+-------+-------+
```

Now we write query–select * from student order by class asc, marks desc;

```
mysql> select * from student order by class asc,marks desc;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      2 | mohak  |     1 |    99 |
|      5 | anil   |     2 |    82 |
|      1 | freya  |    10 |    88 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
+--------+--------+-------+-------+
```

Query result will be ascending order of class and if same class exists then ordering will done on marks column(descending order)

MySQL Group By

The GROUP BY clause groups a set of rows/records into a set of summary rows/records by values of columns or expressions. It returns one row for each group.

We often use the GROUP BY clause with aggregate functions such as SUM, AVG, MAX, MIN, and COUNT. The aggregate function that appears in the SELECT clause provides information about each group.

The GROUP BY clause is an optional clause of the SELECT statement.

Syntax –

```
SELECT   1, c2,..., cn, aggregate_function(ci)
FROM  table WHERE where_conditions GROUP BY c1 , c2,...,cn;
```

Here c1,c2,ci,cn are column name

MySQL Group By

MySQL group by – e.g.

Suppose we are having student table with following data.

```
mysql> select * from student;
+----------+----------+---------+---------+
| rollno   | name     | class   | marks   |
+----------+----------+---------+---------+
|        1 | freya    |      10 |      88 |
|        2 | mohak    |       1 |      99 |
|        3 | vishal   |      10 |      84 |
|        4 | vimal    |      10 |      82 |
|        5 | anil     |       2 |      82 |
+----------+----------+---------+---------+
```

Now we write query–select class from student group by class;

```
mysql> select class from student group by class;
+-------+
| class |
+-------+
|     1 |
|     2 |
|    10 |
+-------+
```

Query result will be unique occurrences of class values,just similar to use distinct clause like(select distinct class from student).

MySQL Group By

MySQL GROUP BY with aggregate functions

The aggregate functions allow us to perform the calculation of a set of rows and return a single value. The GROUP BY clause is often used with an aggregate function to perform calculation and return a single value for each subgroup.

For example, if we want to know the number of student in each class, you can use the COUNT function with the GROUP BY clause as follows:Suppose we are having student table with following data.

```
mysql> select * from student;
+----------+----------+---------+---------+
| rollno   | name     | class   | marks   |
+----------+----------+---------+---------+
|        1 | freya    |      10 |      88 |
|        2 | mohak    |       1 |      99 |
|        3 | vishal   |      10 |      84 |
|        4 | vimal    |      10 |      82 |
|        5 | anil     |       2 |      82 |
+----------+----------+---------+---------+
```

Now we write query–select class,count(*) from student group by class;

```
mysql> select class,count(*) from student group by class;
+-------+----------+
| class | count(*) |
+-------+----------+
|     1 |        1 |
|     2 |        1 |
|    10 |        3 |
+-------+----------+
```

Query result will be unique occurrences of class values along with counting of students(records) of each class(sub group).

MySQL Group By

MySQL GROUP BY with aggregate functions

we are having student table with following data.

```
mysql> select * from student;
+--------+--------+-------+-------+
| rollno | name   | class | marks |
+--------+--------+-------+-------+
|      1 | freya  |    10 |    88 |
|      2 | mohak  |     1 |    99 |
|      3 | vishal |    10 |    84 |
|      4 | vimal  |    10 |    82 |
|      5 | anil   |     2 |    82 |
+--------+--------+-------+-------+
```

Now we write query–select class,avg(marks) from student group by class;

```
mysql> select class,avg(marks) from student group by class;
+-------+-------------+
| class | avg(marks)  |
+-------+-------------+
|     1 |    99.0000 |
|     2 |    82.0000 |
|    10 |    84.6667 |
+-------+-------------+
```

Query result will be unique occurrences of class values along with average marks of each class(sub group).

MySQL Group By

MySQL GROUP BY with aggregate functions (with where and order by clause)

we are having student table with following data.

```
mysql> select * from student;
+--------+---------+-------+--------+
| rollno | name    | class | marks  |
+--------+---------+-------+--------+
|      1 | freya   |    10 |     88 |
|      2 | mohak   |     1 |     99 |
|      3 | vishal  |    10 |     84 |
|      4 | vimal   |    10 |     82 |
|      5 | anil    |     2 |     82 |
+--------+---------+-------+--------+
```

Now we write query–select class,avg(marks) from student where class<10 group by class order by marks desc;

```
mysql> select class,avg(marks) from student where class<10 group by class order
by marks desc;
+-------+------------+
| class | avg(marks) |
+-------+------------+
|     1 |    99.0000 |
|     2 |    82.0000 |
+-------+------------+
```

Query result will be unique occurrences of class values where class<10 along with average marks of each class(sub group) and descending ofer of marks.

MySQL Group by with Having

The HAVING clause is used in the SELECT statement to specify filter conditions for a group of rows or aggregates. The HAVING clause is often used with the GROUP BY clause to filter groups based on a specified condition. To filter the groups returned by GROUP BY clause, we use a HAVING clause.

WHERE is applied before GROUP BY, HAVING is applied after (and can filter on aggregates).

MySQL Group By with having

MySQL GROUP BY with aggregate functions

we are having student table with following data.

```
mysql> select * from student;
+--------+---------+-------+-------+
| rollno | name    | class | marks |
+--------+---------+-------+-------+
|      1 | freya   |    10 |    88 |
|      2 | mohak   |     1 |    99 |
|      3 | vishal  |    10 |    84 |
|      4 | vimal   |    10 |    82 |
|      5 | anil    |     2 |    82 |
+--------+---------+-------+-------+
```

Now we write query–select class,avg(marks) from student group by class having avg(marks)<90;

```
mysql> select class,avg(marks) from student group by class having avg(marks)<90;

+-------+------------+
| class | avg(marks) |
+-------+------------+
|     2 |    82.0000 |
|    10 |    84.6667 |
+-------+------------+
```

Query result will be unique occurrences of class values along with average marks of each class(sub group) and each class having average marks<90.

MySQL Group By with having

MySQL GROUP BY with aggregate functions

we are having student table with following data.

```
mysql> select * from student;
+--------+---------+-------+--------+
| rollno | name    | class | marks  |
+--------+---------+-------+--------+
|      1 | freya   |    10 |     88 |
|      2 | mohak   |     1 |     99 |
|      3 | vishal  |    10 |     84 |
|      4 | vimal   |    10 |     82 |
|      5 | anil    |     2 |     82 |
+--------+---------+-------+--------+
```

Now we write query–select class,avg(marks) from student group by class having count(*)<3;

```
mysql> select class,avg(marks) from student group by class having count(*)<3;
+-------+------------+
| class | avg(marks) |
+-------+------------+
|     1 |    99.0000 |
|     2 |    82.0000 |
+-------+------------+
```

Query result will be unique occurrences of class values along with average marks of each class(sub group) and each class having less than 3 rows.

Operations on relation

RELATIONAL ALGEBRA is a widely used query language and in DBMS concepts. It collects instances of relations as input and gives occurrences of relations as output.

It uses various operations on relation. The output of these operations is a new relation, which might be formed from one or more input relations. Here relation means set or table

Operations on relation

Basic Relational Algebra Operations:

Relational Algebra devided in various groups

Unary Relational Operations

- SELECT (symbol: σ)
- PROJECT (symbol: π)
- RENAME (symbol:)

Relational Algebra Operations From Set Theory

- **UNION (υ)**
- **INTERSECTION ( ),**
- **DIFFERENCE (-)**
- **CARTESIAN PRODUCT (x)**

Binary Relational Operations

- **JOIN**
- DIVISION

We will cover here the highlighted only.

Operations on relation

Union Operator (U)

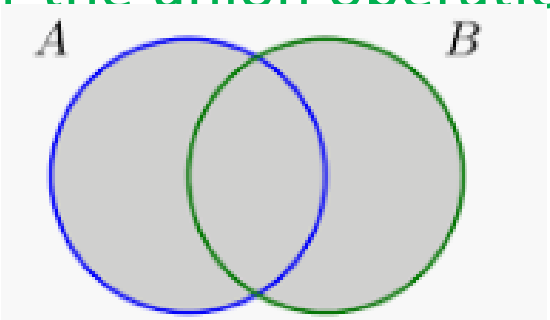
Union operator is denoted by U symbol and it is used to select all the rows (tuples) from two tables (relations).

Lets discuss union operator a bit more. Lets say we have two relations R1 and R2 both have same columns and we want to select all the tuples(rows) from these relations then we can apply the union operator on these relations.

The rows (tuples) that are present in both the tables will only appear once in the union set. In short you can say that there are no duplicates present after the union operation.

Syntax of Union Operator (U)

table_name1 U table_name2



Operations on relation

Union Operator (U) Example

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Freya	S901
C104	Freya	S901
C106	Mohak	S911
C109	praveen	S921
C115	lokesh	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Freya	19
S911	Mohak	18
S921	praveen	19
S931	lokesh	17
S941	chandu	16
S951	rinku	18

Mysql Query

```
SELECT student_name FROM Course
UNION
SELECT Student_name FROM Student
ORDER BY City;
```

Query:

∏ Student_Name (COURSE) U
∏ Student_Name (STUDENT)

Output:

Student_Name

Freya
chandu
praveen
lokesh
rinku
Mohak

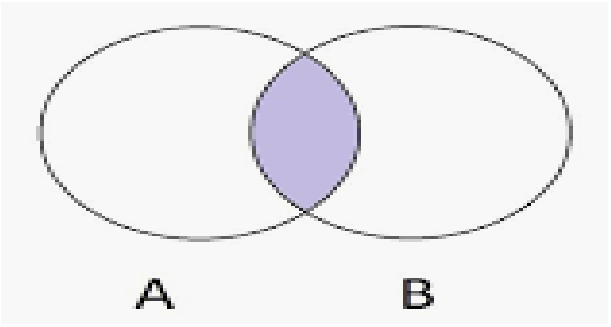
Operations on relation

Intersection Operator (∩)

Intersection operator is denoted by ∩ symbol and it is used to select common rows (tuples) from two tables (relations).

Lets say we have two relations R1 and R2 both have same columns and we want to select all those tuples(rows) that are present in both the relations, then in that case we can apply intersection operation on these two relations R1 ∩ R2.

Only those rows that are present in both the tables will appear in the result set.



Syntax of Intersection Operator (∩)

table_name1 ∩ table_name2

Operations on relation

Intersection Operator (∩) Example

Lets take the same example that we have taken above.

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Freya	S901
C104	Freya	S901
C106	Mohak	S911
C109	praveen	S921
C115	lokesh	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Freya	19
S911	Mohak	18
S921	praveen	19
S931	lokesh	17
S941	chandu	16
S951	rinku	18

Mysql query

```
Select course.student_name from couse ,
student where
course.student_name=student.student_name;
```

Query:

∏ Student_Name (COURSE) ∩
∏ Student_Name (STUDENT)

Output:

Student_Name
Freya
Mohak
praveen
lokesh

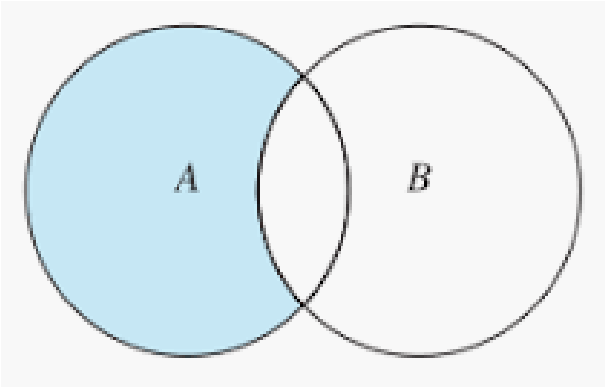
Operations on relation

Minus/Set Difference (-)

Set Difference is denoted by – symbol. Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but not present in Relation R2, this can be done using Set difference R1 – R2.

Syntax of Set Difference (-)

table_name1 - table_name2



Operations on relation

Set Difference (-) Example

Lets take the same example that we have taken above.

Table 1: COURSE

Course_Id	Student_Name	Student_Id
C101	Freya	S901
C104	Freya	S901
C106	Mohak	S911
C109	praveen	S921
C115	lokesh	S931

Table 2: STUDENT

Student_Id	Student_Name	Student_Age
S901	Freya	19
S911	Mohak	18
S921	praveen	19
S931	lokesh	17
S941	chandu	16
S951	rinku	18

Mysql query

```
SELECT c.student_name
FROM student as a
Left joint
course as c on
s.student_name=c.student_name;
```

Mysql does not support minus clause

Query:

Lets write a query to select those student names that are present in STUDENT table but not present in COURSE table.

∏ Student_Name (STUDENT) - ∏ Student_Name (COURSE)

Output:

```
Student_Name
------------
chandu
rinku
```

Operations on relation

Cartesian product (X)/cross joint

Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2.

Operations on relation

Cartesian product (X) example

Table a and Table b as shown below

```
mysql> select * from a;
+---------+-------+
| Name    | val   |
+---------+-------+
| vishal  |    11 |
| ram     |    22 |
+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

Mysql query –

Select * from a,b;

Select * from a cross join b;

```
mysql> select * from a,b;
+---------+------+----------+
| Name    | val  | name     |
+---------+------+----------+
| vishal  |   11 | ram      |
| ram     |   22 | ram      |
| vishal  |   11 | vikrant  |
| ram     |   22 | vikrant  |
+---------+------+----------+
4 rows in set (0.00 sec)
```

Degree of cartesion product is 3 and cardinality is 4=(2 rows of a X 2 rows of b)

Operations on relation

Join – Join is used to fetch data from two or more tables, which is joined to appear as single set of data. It is used for combining column from two or more tables by using values common to both tables.

Types of JOIN

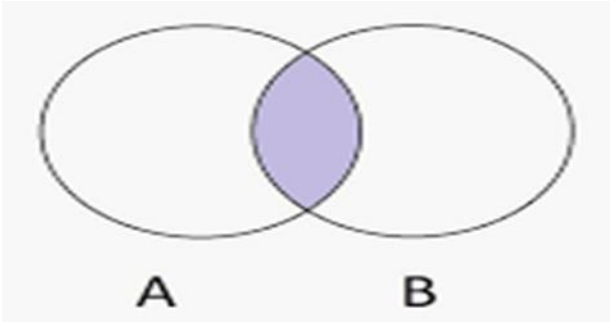
Following are the types of JOIN that we can use in SQL:

- Inner
- Outer
- Left
- Right

Operations on relation

INNER Join or EQUI Join⋈

This is a simple JOIN in which the result is based on matched data as per the equality condition specified in the SQL query.



Operations on relation

INNER Join or EQUI Join example

Table a and Table b as shown below

```
mysql> select * from a;
+---------+-------+
| Name    | val   |
+---------+-------+
| vishal  |    11 |
| ram     |    22 |
+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

Mysql query –

Select course.student_name from couse , student where course.student_name=student.stude nt_name;

Select a.name from a inner join b where a.name=b.name;

```
mysql> select a.name from a inner join b where a.name=b.name;
+------+
| name |
+------+
| ram  |
+------+
```

Operations on relation

Natural JOIN(⋈)

Natural Join is a type of Inner join which is based on column having same name and same datatype present in both the tables to be joined.E.g.

Select * from a natural join b;

```
mysql> select * from a natural join b;
+-------+-------+
| Name  | val   |
+-------+-------+
| ram   |    22 |
+-------+-------+
1 row in set (0.00 sec)
```

Operations on relation

LEFT Outer Join

The left outer join returns a resultset table with the matched data from the two tables and then the remaining rows of the left table and null from the right table's columns. E.g.

```
mysql> select * from a;
+---------+-------+
| Name    | val   |
+---------+-------+
| vishal  |    11 |
| ram     |    22 |
+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

Mysql query –

Select * from a left outer join b on (a.name=b.name);

```
mysql> select * from a left outer join b on (a.name=b.name);
+--------+-----+------+
| Name   | val | name |
+--------+-----+------+
| vishal |  11 | NULL |
| ram    |  22 | ram  |
+--------+-----+------+
2 rows in set (0.02 sec)
```

Operations on relation

RIGHT Outer Join

The right outer join returns a resultset table with the matched data from the two tables being joined, then the remaining rows of the right table and null for the remaining left table's columns.E.g.

```
mysql> select * from a;
+---------+-------+
| Name    | val   |
+---------+-------+
| vishal  |    11 |
| ram     |    22 |
+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

Mysql query –

Select * from a right outer join b on (a.name=b.name);

```
mysql> select * from a right outer join b on (a.name=b.name);
+------+------+---------+
| Name | val  | name    |
+------+------+---------+
| ram  |   22 | ram     |
| NULL | NULL | vikrant |
+------+------+---------+
2 rows in set (0.00 sec)
```

Operations on relation

Full Outer Join ⋈

The full outer join returns a resultset table with the matched data of two table then remaining rows of both left table and then the right table.E.g.

```
mysql> select * from a;
+---------+-------+
| Name    | val   |
+---------+-------+
| vishal  |    11 |
| ram     |    22 |
+---------+-------+
2 rows in set (0.00 sec)

mysql> select * from b;
+----------+
| name     |
+----------+
| ram      |
| vikrant  |
+----------+
2 rows in set (0.00 sec)
```

Mysql query –

Select * from a left outer join b on (a.name=b.name) union Select * from a right outer join b on (a.name=b.name) ;

```
mysql> select * from a left outer join b on(a.name=b.name) union select * from a
 right outer join b on (a.name=b.name);
+--------+------+---------+
| Name   | val  | name    |
+--------+------+---------+
| vishal |   11 | NULL    |
| ram    |   22 | ram     |
| NULL   | NULL | vikrant |
+--------+------+---------+
3 rows in set (0.01 sec)
```